

The Hierarchical Poincaré-Steklov (HPS) solver for elliptic PDEs: A tutorial

P.G. Martinsson, Dept. of Applied Mathematics, University of Colorado at Boulder

June 3, 2015

Abstract: A numerical method for variable coefficient elliptic problems on two dimensional domains is described. The method is based on high-order spectral approximations and is designed for problems with smooth solutions. The resulting system of linear equations is solved using a direct solver with $O(N^{1.5})$ complexity for the pre-computation and $O(N \log N)$ complexity for the solve. The fact that the solver is direct is a principal feature of the scheme, and makes it particularly well suited to solving problems for which iterative solvers struggle; in particular for problems with highly oscillatory solutions. This note is intended as a tutorial description of the scheme, and draws heavily on previously published material.

1. INTRODUCTION

1.1. Problem formulation and outline of solution strategy. This note describes a direct solver for elliptic PDEs with variable coefficients, such as, e.g.,

$$(1) \quad \begin{cases} [Au](\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma, \end{cases}$$

where A is a variable coefficient elliptic differential operator

$$(2) \quad [Au](\mathbf{x}) = -c_{11}(\mathbf{x})[\partial_1^2 u](\mathbf{x}) - 2c_{12}(\mathbf{x})[\partial_1 \partial_2 u](\mathbf{x}) - c_{22}(\mathbf{x})[\partial_2^2 u](\mathbf{x}) \\ + c_1(\mathbf{x})[\partial_1 u](\mathbf{x}) + c_2(\mathbf{x})[\partial_2 u](\mathbf{x}) + c(\mathbf{x}) u(\mathbf{x}),$$

where Ω is a box in the plane with boundary $\Gamma = \partial\Omega$, where all coefficient functions (c , c_i , c_{ij}) are smooth, and where f and g are given functions. (For generalizations, see Section 1.4.) The solver is structured as follows:

- (1) The domain is first tessellated into a hierarchical tree of patches. For each patch on the finest level, a reduced model that we call a “proxy” that represents its internal structure is computed. The proxy takes the form of a dense matrix (that may be stored in a data sparse format), and is for the small patches computed by brute force.
- (2) The larger patches are processed in an upwards pass through the tree. For each patch, its proxy matrix is formed by merging the proxies of its children.
- (3) Once the proxies for all patches have been computed, a solution to the PDE can be computed via a downwards pass through the tree. This step is typically very fast.

We observe that this pattern is similar to the classical nested dissection method of George [5], with a large subsequent literature on “multifrontal solver,” see, e.g., [4, 3] and the references therein.

The techniques described in this note are drawn from [12, 13, 7, 6], which in turn is inspired by earlier work on multidomain spectral methods, see, e.g., [14, 11] and the references therein.

1.2. Dirichlet-to-Neumann maps. In this section, the internal structure of a patch is represented by computing its *Dirichlet-to-Neumann*, or “DtN,” map. To explain what this map does, first observe that for a given boundary function f , the BVP (1) typically has a unique solution ϕ (unless the operator A happens to have a non-trivial null-space, see Remark 1). Now simply form the boundary function h that gives the normal derivative of the solution,

$$h(\mathbf{x}) = \phi_n(\mathbf{x}), \quad \mathbf{x} \in \Gamma,$$

where ϕ_n is the outwards pointing normal derivative. The process for constructing the function h from f is linear, and we write it as

$$h = T f.$$

Or, equivalently,

$$T : \phi|_{\Gamma} \mapsto \phi_n|_{\Gamma}, \quad \text{where } \phi \text{ satisfies } A\phi = 0.$$

In general, the map T is a slightly unpleasant object; it behaves as a differentiation operator, and it has complicated singular behavior near the corners of Γ . A key observation is that in the present context, all these difficulties can be ignored since we limit attention to functions that are smooth. In a sense, we only need to accurately represent the projection of the “true” operator T onto a space of smooth functions (that in particular do not have any corner singularities).

We represent boundary functions by tabulating their values at interpolation nodes on the edges of the boxes. For instance, for a leaf box, we place q Gaussian nodes on each side (for say $q = 20$), which means that the functions f and h are represented by vectors $\mathbf{f}, \mathbf{h} \in \mathbb{R}^{4q}$ and the discrete approximation to T is a $4q \times 4q$ matrix \mathbf{T} . The technique for computing \mathbf{T} for a leaf box is described in Section 3. For a parent box τ with children α and β , there is a technique for computing \mathbf{T}^τ from the matrices \mathbf{T}^α and \mathbf{T}^β that is described in Section 4. In essence, the idea is simply to enforce continuity of both potentials and fluxes across the edge that is shared by Ω_α and Ω_β .

Remark 1. *For a general BVP like (1), the DtN operator need not exist. For example, suppose $-k^2$ is an eigenvalue of Δ with zero Dirichlet data. Then the operator $Au = \Delta u + k^2 u$ clearly has a non-trivial null-space. However, this situation is in some sense “unusual” (since the spectrum of an elliptic PDO on a bounded domain typically is discrete), and it turns out that one can for the most part completely ignore this complication and assume that the DtN operator always exists. Note for instance that if A is coercive (e.g. if $A = -\Delta$), then the DtN is guaranteed to exist for any bounded domain. For problems for which resonances do present numerical problems, there is a variation of the proposed method that is rock-solid stable. The idea is to build a hierarchy of so called impedance maps (instead of DtN maps). These are cousins of the DtN that are defined by*

$$R : (\phi + i\phi_n)|_{\Gamma} \mapsto (\phi - i\phi_n)|_{\Gamma}, \quad \text{where } \phi \text{ satisfies } A\phi = 0.$$

The map R always exists, and is moreover a unitary operator. This construction was proposed by Alex Barnett of Dartmouth College [7].

1.3. Complexity of the direct solver. The asymptotic complexity of the solver described in this note is exactly the same as that for classical nested dissection [5]. For a domain with N interior discretization nodes, the pre-computation (the upwards pass) costs $O(N^{1.5})$ operations, and then the solve (the downwards pass) costs $O(N \log N)$ operations, see [13, Sec. 5.2].

Optimal $O(N)$ complexity can be achieved for both the pre-computation and the solve stage when the Green’s function of the BVP is non-oscillatory. In this case, the matrices \mathbf{T}^τ that approximate the DtN operators have enough internal structure that they can be well represented using a data-sparse format such as, e.g., the \mathcal{H} -matrix framework of Hackbusch and co-workers [9, 8, 1, 2], see [6].

1.4. Generalizations. For notational simplicity, this note treats only the simple Dirichlet problem (1). The scheme can with trivial modifications be applied to more general elliptic operators coupled with Dirichlet, Neumann, or mixed boundary data. It has for instance been successfully tested on convection-diffusion problems that are strongly dominated (by a factor of 10^4) by the convection term. It has also been tested on vibration problems modeled by a variable coefficient Helmholtz equation, and has proven capable of solving problems on domains of size 200×200 wavelengths or more on an office laptop computer. The solutions are computed to seven correct digits. Extension to more general domains is done via parameter maps to a rectangle, or union of rectangles. See [13] for details.

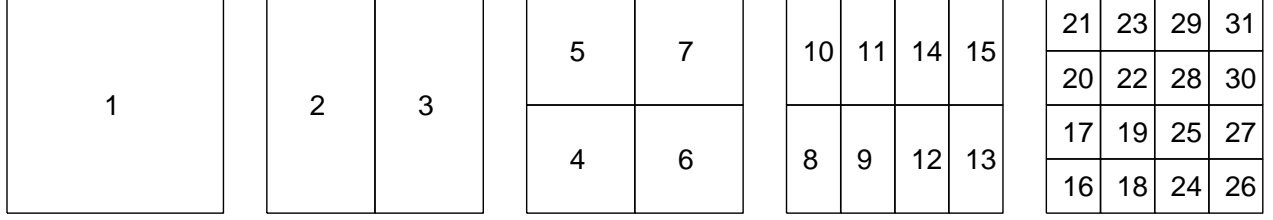


FIGURE 1. The square domain Ω is split into 4×4 leaf boxes. These are then gathered into a binary tree of successively larger boxes as described in Section 5.1. One possible enumeration of the boxes in the tree is shown, but note that the only restriction is that if box τ is the parent of box σ , then $\tau < \sigma$.

1.5. **Outline.** To keep the presentation uncluttered, we start by describing a direct solver for (1) for the case of no body-load, $g = 0$, and with $\Omega = [0, 1]^2$ the unit square. Section 2 introduces the discretization, and how we represent the approximation to the solution u for (1). Section 3 describes the how to compute the DtN operator for a leaf in the tree. Section 4 describes the merge process for how to take the DtN operators for two touching boxes, and computing the DtN operator for their union. Section 5 describes the full hierarchical scheme. Section 6 describes how to solve a problem with body loads.

2. DISCRETIZATION

Partition the domain Ω into a collection of square (or possibly rectangular) boxes, called *leaf boxes*. On the edges of each leaf, place q Gaussian interpolation points. The size of the leaf boxes, and the parameter q should be chosen so that any potential solution u of (1), as well as its first and second derivatives, can be accurately interpolated from their values at these points ($q = 20$ is often a good choice). Let $\{\mathbf{x}_k\}_{k=1}^N$ denote the collection of interpolation points on all boundaries.

Next construct a binary tree on the collection of leaf boxes by hierarchically merging them, making sure that all boxes on the same level are roughly of the same size, cf. Figure 1. The boxes should be ordered so that if τ is a parent of a box σ , then $\tau < \sigma$. We also assume that the root of the tree (i.e. the full box Ω) has index $\tau = 1$. We let Ω_τ denote the domain associated with box τ .

With each box τ , we define two index vectors I_i^τ and I_e^τ as follows:

I_e^τ A list of all *exterior* nodes of τ . In other words, $k \in I_e^\tau$ iff \mathbf{x}_k lies on the boundary of Ω_τ .

I_i^τ For a parent τ , I_i^τ is a list of all its *interior* nodes that are not interior nodes of its children.
For a leaf τ , I_i^τ is empty.

Let $\mathbf{u} \in \mathbb{R}^N$ denote a vector holding approximations to the values of u of (1), in other words,

$$\mathbf{u}(k) \approx u(\mathbf{x}_k).$$

Finally, let $\mathbf{v} \in \mathbb{R}^N$ denote a vector holding approximations to the boundary fluxes of the solution u of (1), in other words

$$\mathbf{v}(k) \approx \begin{cases} \partial_2 u(\mathbf{x}_k), & \text{when } \mathbf{x}_j \text{ lies on a horizontal edge,} \\ \partial_1 u(\mathbf{x}_k), & \text{when } \mathbf{x}_j \text{ lies on a vertical edge.} \end{cases}$$

Note the sign convention for the normal derivatives: we use a global frame of reference, as opposed to distinguishing between outwards and inwards pointing normal derivatives. This is a deliberate choice to avoid problems with signs when matching fluxes of touching boxes.

3. CONSTRUCTING THE DIRICHLET-TO-NEUMANN MAP FOR A LEAF

This section describes a spectral method for computing a discrete approximation to the DtN map T^τ associated with a leaf box Ω_τ . In other words, if u is a solution of (1), we seek a matrix \mathbf{T}^τ of size $4q \times 4q$ such that

$$(3) \quad \begin{array}{ccc} \mathbf{v}(I_e^\tau) & \approx & \mathbf{T}^\tau \mathbf{u}(I_e^\tau) \\ \text{Neumann data} & & \text{DtN map} \quad \text{Dirichlet data} \end{array}$$

Conceptually, we proceed as follows: Given a vector $\mathbf{u}(I_e^\tau) \in \mathbb{R}^{4q}$ specifying the solution u on the boundary of Ω_τ , form for each side the unique polynomial of degree at most $q - 1$ that interpolates the q specified values of u . This yields Dirichlet boundary data on Ω_τ in the form of four polynomials. Solve the restriction of (1) to Ω_τ for the specified boundary data using a spectral method on a local tensor product grid of $p \times p$ Chebyshev nodes (typically, we choose $p = q + 1$). The vector $\mathbf{v}(I_e^\tau)$ is obtained by spectral differentiation of the local solution, and then re-tabulating the boundary fluxes to the Gaussian nodes in $\{\mathbf{x}_k\}_{k \in I_e^\tau}$.

We give details of the construction in Section 3.2, but as a preliminary step, we first review a classical spectral collocation method for the local solve in Section 3.1

Remark 2. Chebyshev nodes are ideal for the leaf computations, and it is in principle also possible to use Chebyshev nodes to represent all boundary-to-boundary “solution operators” such as, e.g., \mathbf{T}^τ (indeed, this was the approach taken in the first implementation of the proposed method [13]). However, there are at least two substantial benefits to using Gaussian nodes that justify the trouble to re-tabulate the operators. First, the procedure for merging boundary operators defined for neighboring boxes is much cleaner and involves less bookkeeping since the Gaussian nodes do not include the corner nodes. (Contrast Section 4 of [13] with Section 4.) Second, and more importantly, the use of the Gaussian nodes allows for interpolation between different discretizations. Thus the method can easily be extended to have local refinement when necessary, see Remark 4.

3.1. Spectral discretization. Let Ω_τ denote a rectangular subset of Ω with boundary Γ_τ , and consider the local Dirichlet problem with the boundary data given by a “dummy” function ψ

$$(4) \quad \begin{cases} [Au](\mathbf{x}) = 0, & \mathbf{x} \in \Omega_\tau, \\ u(\mathbf{x}) = \psi(\mathbf{x}), & \mathbf{x} \in \Gamma_\tau, \end{cases}$$

where the elliptic operator A is defined by (2). We will construct an approximate solution to (4) using a classical spectral collocation method described in, e.g., Trefethen [15]: First, pick a small integer p and let $\{\mathbf{z}_k\}_{k=1}^{p^2}$ denote the nodes in a tensor product grid of $p \times p$ Chebyshev nodes on Ω_τ . Let $\mathbf{D}^{(1)}$ and $\mathbf{D}^{(2)}$ denote spectral differentiation matrices corresponding to the operators $\partial/\partial x_1$ and $\partial/\partial x_2$, respectively. The operator (2) is then locally approximated via the $p^2 \times p^2$ matrix

$$(5) \quad \mathbf{A} = -\mathbf{C}_{11}(\mathbf{D}^{(1)})^2 - 2\mathbf{C}_{12}\mathbf{D}^{(1)}\mathbf{D}^{(2)} - \mathbf{C}_{22}(\mathbf{D}^{(2)})^2 + \mathbf{C}_1\mathbf{D}^{(1)} + \mathbf{C}_2\mathbf{D}^{(2)} + \mathbf{C},$$

where \mathbf{C}_{11} is the diagonal matrix with diagonal entries $\{c_{11}(\mathbf{z}_k)\}_{k=1}^{p^2}$, and the other matrices \mathbf{C}_{ij} , \mathbf{C}_i , \mathbf{C} are defined analogously.

Let $\mathbf{w} \in \mathbb{R}^{p^2}$ denote a vector holding the desired approximate solution of (4). We populate all entries corresponding to boundary nodes with the Dirichlet data from ψ , and then enforce a spectral collocation condition at the interior nodes. To formalize, let us partition the index set

$$\{1, 2, \dots, p^2\} = J_e \cup J_i$$

in such a way that J_e contains the $4(p - 1)$ nodes on the boundary of Ω_τ , and J_i denotes the set of $(p - 2)^2$ interior nodes, see Figure 2(a). Then partition the vector \mathbf{w} into two parts corresponding to internal and exterior nodes via

$$\mathbf{w}_i = \mathbf{w}(J_i), \quad \mathbf{w}_e = \mathbf{w}(J_e).$$

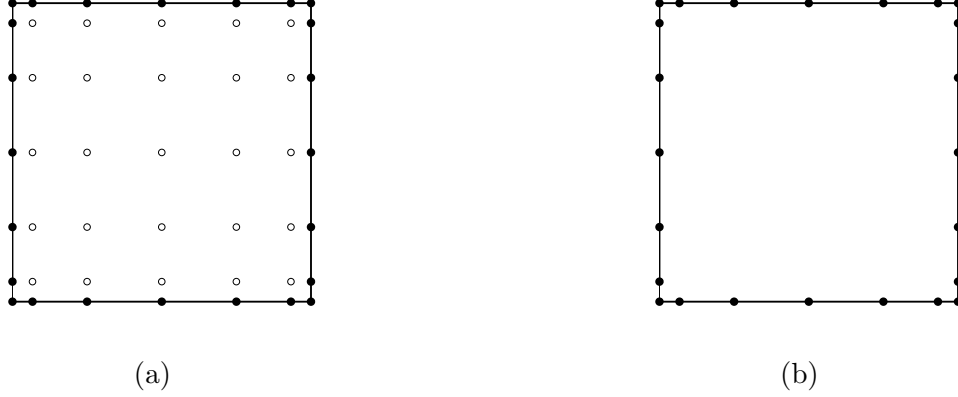


FIGURE 2. Notation for the leaf computation in Section 3. (a) A leaf before elimination of interior (white) nodes. (b) A leaf after elimination of interior nodes.

Analogously, partition \mathbf{A} into four parts via

$$\mathbf{A}_{i,i} = \mathbf{A}(J_i, J_i), \quad \mathbf{A}_{i,e} = \mathbf{A}(J_i, J_e), \quad \mathbf{A}_{e,i} = \mathbf{A}(J_e, J_i), \quad \mathbf{A}_{e,e} = \mathbf{A}(J_e, J_e).$$

The potential at the exterior nodes is now given directly from the boundary condition:

$$\mathbf{w}_e = [\psi(\mathbf{z}_k)]_{k \in J_e}.$$

For the internal nodes, we enforce the PDE (4) via direct collocation:

$$(6) \quad \mathbf{A}_{i,i} \mathbf{w}_i + \mathbf{A}_{i,e} \mathbf{w}_e = \mathbf{0}.$$

Solving (6) for \mathbf{w}_i , we find

$$(7) \quad \mathbf{w}_i = -\mathbf{A}_{i,i}^{-1} \mathbf{A}_{i,e} \mathbf{w}_e,$$

3.2. Constructing the approximate DtN. Now that we know how to approximately solve the local Dirichlet problem (4) via a local spectral method, we can build a matrix \mathbf{T}^τ such that (3) holds to high accuracy. The starting point is a vector $\mathbf{u}(I_\tau) \in \mathbb{R}^{4q}$ of tabulated potential values on the boundary of Ω_τ . We will construct the vector $\mathbf{v}(I_\tau) \in \mathbb{R}^{4q}$ via four linear maps. The combination of these maps is the matrix \mathbf{T}^τ .

Step 1 — re-tabulation from Gaussian nodes to Chebyshev nodes: For each side of Ω_τ , form the unique interpolating polynomial of degree at most $q - 1$ that interpolates the q potential values on that side specified by $\mathbf{u}(I_\tau)$. Now evaluate these polynomials at the boundary nodes of a $p \times p$ Chebyshev grid on Ω_τ . Observe that for a corner node, we may in the general case get conflicts. For instance, the potential at the south-west corner may get one value from extrapolation of potential values on the south border, and one value from extrapolation of the potential values on the west border. We resolve such conflicts by assigning the corner node the average of the two possibly different values. (In practice, essentially no error occurs since we know that the vector $\mathbf{u}(I_\tau)$ tabulates an underlying function that is continuous at the corner.)

Step 2 — spectral solve: Step 1 populates the boundary nodes of the $p \times p$ Chebyshev grid with Dirichlet data. Now determine the potential at all interior points on the Chebyshev grid by executing a local spectral solve, cf. equation (7).

Step 3 — spectral differentiation: After Step 2, the potential is known at all nodes on the local Chebyshev grid. Now perform spectral differentiation to evaluate approximations to $\partial u / \partial x_2$

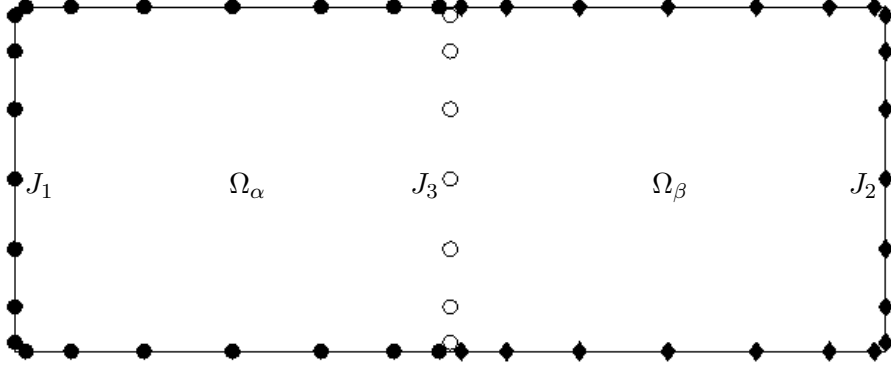


FIGURE 3. Notation for the merge operation described in Section 4. The rectangular domain Ω is formed by two squares Ω_α and Ω_β . The sets J_1 and J_2 form the exterior nodes (black), while J_3 consists of the interior nodes (white).

for the Chebyshev nodes on the two horizontal sides, and $\partial u / \partial x_1$ for the Chebyshev nodes on the two vertical sides.

Step 4 — re-tabulation from the Chebyshev nodes back to Gaussian nodes: After Step 3, the boundary fluxes on $\partial\Omega_\tau$ are specified by four polynomials of degree $p-1$ (specified via tabulation on the relevant Chebyshev nodes). Now simply evaluate these polynomials at the Gaussian nodes on each side to obtain the vector $\mathbf{v}(I_e^\tau)$.

Putting everything together, we find that the matrix \mathbf{T}^τ is given as a product of four matrices

$$\begin{array}{ccccccc} \mathbf{T}^\tau & = & \mathbf{L}_4 & \circ & \mathbf{L}_3 & \circ & \mathbf{L}_2 & \circ & \mathbf{L}_1 \\ 4q \times 4q & & 4q \times 4p & & 4p \times p^2 & & p^2 \times 4(p-1) & & 4(p-1) \times 4q \end{array}$$

where \mathbf{L}_i is the linear transform corresponding to “Step i ” above. Observe that many of these transforms are far from dense, for instance, \mathbf{L}_1 and \mathbf{L}_4 are 4×4 block matrices with all off-diagonal blocks equal to zero. Exploiting these structures substantially accelerates the computation.

4. MERGING TWO DTN MAPS

Let τ denote a box in the tree with children α and β . In this section, we demonstrate that if the DtN matrices \mathbf{T}^α and \mathbf{T}^β for the children are known, then the DtN matrix \mathbf{T}^τ can be constructed via a purely local computation which we refer to as a “merge” operation.

We start by introducing some notation: Let Ω_τ denote a box with children Ω_α and Ω_β . For concreteness, let us assume that Ω_α and Ω_β share a vertical edge as shown in Figure 3, so that

$$\Omega_\tau = \Omega_\alpha \cup \Omega_\beta.$$

We partition the points on $\partial\Omega_\alpha$ and $\partial\Omega_\beta$ into three sets:

- J_1 Boundary nodes of Ω_α that are not boundary nodes of Ω_β .
- J_2 Boundary nodes of Ω_β that are not boundary nodes of Ω_α .
- J_3 Boundary nodes of both Ω_α and Ω_β that are *not* boundary nodes of the union box Ω_τ .

Figure 3 illustrates the definitions of the J_k ’s. Let u denote a solution to (1), with tabulated potential values \mathbf{u} and boundary fluxes \mathbf{v} , as described in Section 2. Set

$$(8) \quad \mathbf{u}_i = \mathbf{u}_3, \quad \text{and} \quad \mathbf{u}_e = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}.$$

Recall that \mathbf{T}^α and \mathbf{T}^β denote the operators that map values of the potential u on the boundary to values of $\partial_n u$ on the boundaries of the boxes Ω_α and Ω_β , as described in Section 3. The operators can be partitioned according to the numbering of nodes in Figure 3, resulting in the equations

$$(9) \quad \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{1,1}^\alpha & \mathbf{T}_{1,3}^\alpha \\ \mathbf{T}_{3,1}^\alpha & \mathbf{T}_{3,3}^\alpha \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_3 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{2,2}^\beta & \mathbf{T}_{2,3}^\beta \\ \mathbf{T}_{3,2}^\beta & \mathbf{T}_{3,3}^\beta \end{bmatrix} \begin{bmatrix} \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}.$$

Our objective is now to construct a solution operator \mathbf{S}^τ and a DtN matrix \mathbf{T}^τ such that

$$(10) \quad \mathbf{u}_3 = \mathbf{S}^\tau \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}$$

$$(11) \quad \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} = \mathbf{T}^\tau \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}.$$

To this end, we eliminate \mathbf{v}_3 from (9) and write the result as a single equation:

$$(12) \quad \left[\begin{array}{cc|c} \mathbf{T}_{1,1}^\alpha & \mathbf{0} & \mathbf{T}_{1,3}^\alpha \\ \mathbf{0} & \mathbf{T}_{2,2}^\beta & \mathbf{T}_{2,3}^\beta \\ \hline \mathbf{T}_{3,1}^\alpha & -\mathbf{T}_{3,2}^\beta & \mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta \end{array} \right] \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{0} \end{bmatrix},$$

The last equation directly tells us that (10) holds with

$$(13) \quad \mathbf{S}^\tau = (\mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta)^{-1} [-\mathbf{T}_{3,1}^\alpha \mid \mathbf{T}_{3,2}^\beta].$$

By eliminating \mathbf{u}_3 from (12) by forming a Schur complement, we also find that (11) holds with

$$(14) \quad \mathbf{T}^\tau = \begin{bmatrix} \mathbf{T}_{1,1}^\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{2,2}^\beta \end{bmatrix} + \begin{bmatrix} \mathbf{T}_{1,3}^\alpha \\ \mathbf{T}_{2,3}^\beta \end{bmatrix} (\mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta)^{-1} [-\mathbf{T}_{3,1}^\alpha \mid \mathbf{T}_{3,2}^\beta].$$

5. THE FULL HIERARCHICAL SCHEME

At this point, we know how to construct the DtN operator for a leaf (Section 3), and how to merge two such operators of neighboring patches to form the DtN operator of their union (Section 4). We are ready to describe the full hierarchical scheme for solving the Dirichlet problem (1). This scheme takes the Dirichlet boundary data f , and constructs an approximation to the solution u . The output is a vector \mathbf{u} that tabulates approximations to u at the Gaussian nodes $\{\mathbf{x}_k\}_{k=1}^N$ on all interior edges that were defined in Section 2. To find u at an arbitrary set of target points in Ω , a post-processing step described in Section 5.3 can be used.

5.1. The algorithm. Partition the domain into a hierarchical tree as described in Section 2. Then execute a “build stage” in which we construct for each box τ the following two matrices:

\mathbf{S}^τ For a parent box τ , \mathbf{S}^τ is a solution operator that maps values of u on $\partial\Omega_\tau$ to values of u at the interior nodes. In other words, $\mathbf{u}(I_i^\tau) = \mathbf{S}^\tau \mathbf{u}(I_e^\tau)$. (For a leaf τ , \mathbf{S}^τ is not defined.)

\mathbf{T}^τ The matrix that maps $\mathbf{u}(I_e^\tau)$ (tabulating values of u on $\partial\Omega_\tau$) to $\mathbf{v}(I_e^\tau)$ (tabulating values of du/dn). In other words, $\mathbf{v}(I_e^\tau) = \mathbf{T}^\tau \mathbf{u}(I_e^\tau)$.

(Recall that the index vectors I_e^τ and I_i^τ were defined in Section 2.) The build stage consists of a single sweep over all nodes in the tree. Any bottom-up ordering in which any parent box is processed after its children can be used. For each leaf box τ , an approximation to the local DtN map \mathbf{T}^τ is constructed using the procedure described in Section 3. For a parent box τ with children α and β , the matrices \mathbf{S}^τ and \mathbf{T}^τ are formed from the DtN operators \mathbf{T}^α and \mathbf{T}^β via the process described in Section 4. Algorithm 1 summarizes the build stage.

ALGORITHM 1 (build solution operators)

This algorithm builds the global Dirichlet-to-Neumann operator for (1). It also builds all matrices \mathbf{S}^τ required for constructing u at any interior point. It is assumed that if node τ is a parent of node σ , then $\tau < \sigma$.

-
- (1) **for** $\tau = N_{\text{boxes}}, N_{\text{boxes}} - 1, N_{\text{boxes}} - 2, \dots, 1$
 - (2) **if** (τ is a leaf)
 - (3) Construct \mathbf{T}^τ via the process described in Section 3.
 - (4) **else**
 - (5) Let α and β be the children of τ .
 - (6) Split I_e^α and I_e^β into vectors I_1, I_2 , and I_3 as shown in Figure 3.
 - (7) $\mathbf{S}^\tau = (\mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta)^{-1} [-\mathbf{T}_{3,1}^\alpha \mid \mathbf{T}_{3,2}^\beta]$
 - (8) $\mathbf{T}^\tau = \begin{bmatrix} \mathbf{T}_{1,1}^\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{2,2}^\beta \end{bmatrix} + \begin{bmatrix} \mathbf{T}_{1,3}^\alpha \\ \mathbf{T}_{2,3}^\beta \end{bmatrix} \mathbf{S}^\tau.$
 - (9) Delete \mathbf{T}^α and \mathbf{T}^β .
 - (10) **end if**
 - (11) **end for**
-

ALGORITHM 2 (solve BVP once solution operator has been built)

This program constructs an approximation \mathbf{u} to the solution u of (1). It assumes that all matrices \mathbf{S}^τ have already been constructed in a pre-computation. It is assumed that if node τ is a parent of node σ , then $\tau < \sigma$.

-
- (1) $\mathbf{u}(k) = f(\mathbf{x}_k)$ for all $k \in I_e^1$.
 - (2) **for** $\tau = 1, 2, 3, \dots, N_{\text{boxes}}$
 - (3) **if** (τ is a parent)
 - (4) $\mathbf{u}(I_i^\tau) = \mathbf{S}^\tau \mathbf{u}(I_e^\tau).$
 - (5) **end if**
 - (6) **end for**
-

Remark: This algorithm outputs the solution on the Gaussian nodes on box boundaries. To get the solution at other points, use the method described in Section 5.3.

Once all the matrices $\{\mathbf{S}^\tau\}_\tau$ have been formed, a vector \mathbf{u} holding approximations to the solution u of (1) can be constructed for all discretization points by starting at the root box Ω and moving down the tree toward the leaf boxes. The values of \mathbf{u} for the points on the boundary of Ω can be obtained by tabulating the boundary function f . When any box τ is processed, the value of \mathbf{u} is known for all nodes on its boundary (i.e. those listed in I_e^τ). The matrix \mathbf{S}^τ directly maps these values to the values of \mathbf{u} on the nodes in the interior of τ (i.e. those listed in I_i^τ). When all nodes have been processed, approximations to u have constructed for all tabulation nodes on interior edges. Algorithm 2 summarizes the solve stage.

Remark 3. *The merge stage is exact when performed in exact arithmetic. The only approximation involved is the approximation of the solution u on a leaf by its interpolating polynomial.*

Remark 4. *To keep the presentation simple, we consider in this note only the case of a uniform computational grid. Such grids are obviously not well suited to situations where the regularity of the solution changes across the domain. The method described can in principle be modified to handle locally refined grids quite easily. A complication is that the tabulation nodes for two touching boxes will typically not coincide, which requires the introduction of specialized interpolation operators. Efficient*

refinement strategies also require the development of error indicators that identify the regions where the grid need to be refined. This is work in progress, and will be reported at a later date. We observe that our introduction of Gaussian nodes on the internal boundaries (as opposed to the Chebyshev nodes used in [13]) makes re-interpolation much easier.

5.2. Asymptotic complexity. In this section, we determine the asymptotic complexity of the direct solver. Let $N_{\text{leaf}} = 4q$ denote the number of Gaussian nodes on the boundary of a leaf box, and let q^2 denote the number of Chebychev nodes used in the leaf computation. Let L denote the number of levels in the binary tree. This means there are 4^L boxes. Thus the total number of discretization nodes N is approximately $4^L q = \frac{(2^L q^2)}{q}$. (To be exact, $N = 2^{2L+1}q + 2^{L+1}q$.)

The cost to process one leaf is approximately $O(q^6)$. Since there are $\frac{N}{q^2}$ leaf boxes, the total cost of pre-computing approximate DtN operators for all the bottom level is $\frac{N}{q^2} \times q^6 \sim Nq^4$.

Next, consider the cost of constructing the DtN map on level ℓ via the merge operation described in Section 4. For each box on the level ℓ , the operators \mathbf{T}^τ and \mathbf{S}^τ are constructed via (13) and (13). These operations involve matrices of size roughly $2^{-\ell}N^{0.5} \times 2^{-\ell}N^{0.5}$. Since there are 4^ℓ boxes per level. The cost on level ℓ of the merge is

$$4^\ell \times \left(2^{-\ell}N^{0.5}\right)^3 \sim 2^{-\ell}N^{1.5}.$$

The total cost for all the merge procedures has complexity

$$\sum_{\ell=1}^L 2^{-\ell}N^{1.5} \sim N^{1.5}.$$

Finally, consider the cost of the downwards sweep which solves for the interior unknowns. For any non-leaf box τ on level ℓ , the size of \mathbf{S}^τ is $2^l q \times 2^l(6q)$ which is approximately $\sim 2^{-\ell}N^{0.5} \times 2^{-\ell}N^{0.5}$. Thus the cost of applying \mathbf{S}^τ is roughly $(2^{-\ell}N^{0.5})^2 = 2^{-2\ell}N$. So the total cost of the solve step has complexity

$$\sum_{l=0}^{L-1} 2^{2l} 2^{-2l} N \sim N \log N.$$

In [6], we explain how to exploit structure in the matrices \mathbf{T} and \mathbf{S} to improve the computational cost of both the precomputation and the solve steps.

5.3. Post-processing. The direct solver in Algorithm 1 constructs approximations to the solution u of (1) at tabulation nodes at all interior edges. Once these are available, it is easy to construct an approximation to u at an arbitrary point. To illustrate the process, suppose that we seek an approximation to $u(\mathbf{y})$, where \mathbf{y} is a point located in a leaf τ . We have values of u tabulated at Gaussian nodes on $\partial\Omega_\tau$. These can easily be re-interpolated to the Chebyshev nodes on $\partial\Omega_\tau$. Then u can be reconstructed at the interior Chebyshev nodes via the formula (7); observe that the local solution operator $-\mathbf{A}_{i,i}^{-1}\mathbf{A}_{i,e}$ was built when the leaf was originally processed and can be simply retrieved from memory (assuming enough memory is available). Once u is tabulated at the Chebyshev grid on Ω_τ , it is trivial to interpolate it to \mathbf{y} or any other point.

6. BODY LOADS

Now that we have described how to solve our basic boundary value problem

$$(15) \quad \begin{cases} [Au](\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma, \end{cases}$$

for the special case where $g = 0$, we will next consider the more general case that includes a body load. Only minor modifications are required to the basic scheme, but note that the resulting method requires substantially more memory. The techniques presented here were developed jointly with Tracy Babb of CU-Boulder, cf. [10].

6.1. Notation. When handling body loads, we will extensively switch between the Chebyshev and Gaussian grids, so we need to introduce some additional notation.

Let $\{\mathbf{y}_j\}_{j=1}^M$ denote the global grid obtained by putting down a $p \times p$ tensor product grid of Chebyshev nodes on each leaf. For a leaf τ , let I_c^τ denote an index vector pointing to the nodes in $\{\mathbf{y}_j\}_{j=1}^M$ that lie on leaf τ . We partition this index vector into exterior and interior nodes as follows

$$I_c^\tau = I_{ce}^\tau \cup I_{ci}^\tau.$$

To avoid confusion with the index vectors pointing into the grid of Gaussian boundary functions, we rename these index vectors as follows:

- I_{ge}^τ : An index vector marking the (Gauss) nodes in $\{\mathbf{x}_i\}_{i=1}^N$ that lie on $\partial\Omega_\tau$.
- I_{gi}^τ : For a parent node τ , this is an index vector marking the (Gauss) nodes in $\{\mathbf{x}_i\}_{i=1}^N$ that lie on the “interior” boundary of τ . For a leaf node, this vector is not defined.

The fact that we use two spectral grids also leads to a need to distinguish between different vectors tabulating approximate values. We have:

	τ is a leaf	τ is a parent
\mathbf{u}_c^τ	u tabulated on Chebyshev nodes	—
\mathbf{u}_{ce}^τ	u tabulated on Chebyshev exterior nodes	—
\mathbf{u}_{ci}^τ	u tabulated on Chebyshev interior nodes	—
\mathbf{u}_{ge}^τ	u tabulated on Gaussian exterior nodes	u tabulated on Gaussian exterior nodes
\mathbf{u}_{gi}^τ	—	u tabulated on Gaussian interior nodes

6.2. Leaf computation. Let τ be a leaf. We split the solution u to the equation

$$(16) \quad \begin{cases} Au(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega_\tau, \\ u(\mathbf{x}) = \psi(\mathbf{x}), & \mathbf{x} \in \Gamma_\tau, \end{cases}$$

as

$$u = w + \phi$$

where w is a *particular solution*

$$(17) \quad \begin{cases} Aw(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega_\tau, \\ w(\mathbf{x}) = 0, & \mathbf{x} \in \Gamma_\tau, \end{cases}$$

and where ϕ is a *homogeneous solution*

$$(18) \quad \begin{cases} A\phi(\mathbf{x}) = 0, & \mathbf{x} \in \Omega_\tau, \\ \phi(\mathbf{x}) = \psi(\mathbf{x}), & \mathbf{x} \in \Gamma_\tau. \end{cases}$$

We can now write the Neumann data for u as

$$\partial_n u|_{\Gamma_\tau} = \partial_n w|_{\Gamma_\tau} + \partial_n \phi|_{\Gamma_\tau} = \partial_n w|_{\Gamma_\tau} + T\phi|_{\Gamma_\tau} = \partial_n w|_{\Gamma_\tau} + T\psi,$$

where, as before, T is the NfD operator. Our objective is therefore to find a matrix that maps the given body load g on Ω_τ to the Neumann data of w . We do this calculation on the Chebyshev grid. Discretizing (17), and collocating on the internal nodes, we find

$$\mathbf{A}_{\text{ci,ce}} \mathbf{w}_{\text{ce}} + \mathbf{A}_{\text{ci,ci}} \mathbf{w}_{\text{ci}} = \mathbf{g}_{\text{ci}}.$$

But now observe that $\mathbf{w}_{\text{ce}} = 0$, so the particular solution is given by

$$\mathbf{w}_{\text{c}} = \begin{bmatrix} \mathbf{w}_{\text{ce}} \\ \mathbf{w}_{\text{ci}} \end{bmatrix} = \mathbf{F}_{\text{c,ci}} \mathbf{g}_{\text{ci}}, \quad \text{where} \quad \mathbf{F}_{\text{c,ci}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{A}_{\text{ci,ci}}^{-1} \end{bmatrix}.$$

Let \mathbf{h}_{ge} denote the Neumann data for \mathbf{w} , tabulated at the Gaussian nodes on the boundary. It follows that

$$\mathbf{h}_{\text{ge}} = \underbrace{\mathbf{D}_{\text{ge,c}} \mathbf{F}_{\text{c,ci}}}_{=:\mathbf{H}_{\text{ge,ci}}} \mathbf{g}_{\text{ci}},$$

where $\mathbf{D}_{\text{ge,c}}$ is the operator that differentiates from the full Chebyshev grid, and then interpolates to the Gaussian exterior nodes. (Using the notation from Section 4, $\mathbf{D}_{\text{ge,c}} = \mathbf{L}_4 \mathbf{L}_3$.)

Once the boundary data of u on Ω_τ is given, the total solution on the Chebyshev grid is given by

$$\mathbf{u}_{\text{c}} = \mathbf{S}_{\text{c,ge}} \mathbf{u}_{\text{ge}} + \mathbf{F}_{\text{c,ci}} \mathbf{g}_{\text{ci}} = \mathbf{S}_{\text{c,ge}} \mathbf{u}_{\text{ge}} + \mathbf{w}_{\text{c}},$$

where $\mathbf{S}_{\text{c,ge}}$ is the solution operator given by

$$\mathbf{S}_{\text{c,ge}} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{A}_{\text{ci,ci}}^{-1} \mathbf{A}_{\text{ci,ce}} \end{bmatrix} \mathbf{I}_{\text{ce,ge}},$$

where, in turn, $\mathbf{I}_{\text{ce,ge}}$ is the interpolation operator that maps from the boundary Gauss nodes to the boundary Chebyshev nodes.

6.3. Merge operator. Let τ be a parent node with children α and β . The local equilibrium equations read

$$(19) \quad \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{1,1}^\alpha & \mathbf{T}_{1,3}^\alpha \\ \mathbf{T}_{3,1}^\alpha & \mathbf{T}_{3,3}^\alpha \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_3 \end{bmatrix} + \begin{bmatrix} \mathbf{h}_1^\alpha \\ \mathbf{h}_3^\alpha \end{bmatrix},$$

$$(20) \quad \begin{bmatrix} \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{2,2}^\beta & \mathbf{T}_{2,3}^\beta \\ \mathbf{T}_{3,2}^\beta & \mathbf{T}_{3,3}^\beta \end{bmatrix} \begin{bmatrix} \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix} + \begin{bmatrix} \mathbf{h}_2^\beta \\ \mathbf{h}_3^\beta \end{bmatrix}.$$

Combine the two equations for \mathbf{v}_3 to obtain the equation

$$\mathbf{T}_{3,1}^\alpha \mathbf{u}_1 + \mathbf{T}_{3,3}^\alpha \mathbf{u}_3 + \mathbf{h}_3^\alpha = \mathbf{T}_{3,2}^\beta \mathbf{u}_2 + \mathbf{T}_{3,3}^\beta \mathbf{u}_3 + \mathbf{h}_3^\beta.$$

This gives [check for sign errors!]

$$(21) \quad \mathbf{u}_3 = (\mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta)^{-1} (\mathbf{T}_{3,2}^\beta \mathbf{u}_2 - \mathbf{T}_{3,1}^\alpha \mathbf{u}_1 + \mathbf{h}_3^\beta - \mathbf{h}_3^\alpha)$$

Inserting (21) back into (19) we find

$$\begin{aligned} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} &= \left(\begin{bmatrix} \mathbf{T}_{1,1}^\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{2,2}^\beta \end{bmatrix} + \begin{bmatrix} \mathbf{T}_{1,3}^\alpha \\ \mathbf{T}_{2,3}^\beta \end{bmatrix} (\mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta)^{-1} [-\mathbf{T}_{3,1}^\alpha \mid \mathbf{T}_{3,2}^\beta] \right) \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} + \\ &\quad \begin{bmatrix} \mathbf{h}_1^\alpha \\ \mathbf{h}_2^\beta \end{bmatrix} + \begin{bmatrix} \mathbf{T}_{1,3}^\alpha \\ \mathbf{T}_{2,3}^\beta \end{bmatrix} (\mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta)^{-1} (\mathbf{h}_3^\beta - \mathbf{h}_3^\alpha). \end{aligned}$$

We now define operators

$$\begin{aligned}\mathbf{X}_{\text{gi,gi}}^\tau &= (\mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta)^{-1}, \\ \mathbf{S}_{\text{gi,ge}}^\tau &= (\mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta)^{-1} [-\mathbf{T}_{3,1}^\alpha \mid \mathbf{T}_{3,2}^\beta] = \mathbf{X}_{\text{gi,gi}}^\tau [-\mathbf{T}_{3,1}^\alpha \mid \mathbf{T}_{3,2}^\beta], \\ \mathbf{T}_{\text{ge,ge}}^\tau &= \begin{bmatrix} \mathbf{T}_{1,1}^\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{2,2}^\beta \end{bmatrix} (\mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta)^{-1} [-\mathbf{T}_{3,1}^\alpha \mid \mathbf{T}_{3,2}^\beta] = \begin{bmatrix} \mathbf{T}_{1,1}^\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{2,2}^\beta \end{bmatrix} \mathbf{S}_{\text{gi,ge}}^\tau.\end{aligned}$$

Then in the upwards pass in the solve, we compute

$$\begin{aligned}\mathbf{w}_{\text{gi}}^\tau &= \mathbf{X}^\tau (\mathbf{h}_3^\beta - \mathbf{h}_3^\alpha) \\ \mathbf{h}_{\text{ge}}^\tau &= \begin{bmatrix} \mathbf{T}_{1,3}^\alpha \\ \mathbf{T}_{2,3}^\beta \end{bmatrix} \mathbf{w}_{\text{gi}}^\tau,\end{aligned}$$

and in the solve stage, we recover $\mathbf{u}_{\text{gi}}^\tau$ via

$$\mathbf{u}_{\text{gi}}^\tau = \mathbf{S}_{\text{gi,ge}} \mathbf{u}_{\text{ge}}^\tau + \mathbf{w}_{\text{gi}}^\tau.$$

Remark 5 (Physical interpretation of merge). *The quantities $\mathbf{w}_{\text{gi}}^\tau$ and $\mathbf{h}_{\text{ge}}^\tau$ have a simple physical meaning. The vector $\mathbf{w}_{\text{gi}}^\tau$ introduced above is simply a tabulation of the particular solution w^τ associated with τ on the interior boundary Γ_3 , and $\mathbf{h}_{\text{ge}}^\tau$ is the normal derivative of w^τ . To be precise, w^τ is the solution to the inhomogeneous problem, cf. (17)*

$$(22) \quad \begin{cases} Aw^\tau(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega_\tau, \\ w^\tau(\mathbf{x}) = 0, & \mathbf{x} \in \Gamma_\tau. \end{cases}$$

We can re-derive the formula for $w|_{\Gamma_3}$ using the original mathematical operators as follows: First observe that for $\mathbf{x} \in \Omega^\alpha$, we have $A(w^\tau - w^\alpha) = g - g = 0$, so the NfD operator T^α applies to the function $w^\tau - w^\alpha$:

$$T_{31}^\alpha(w_1^\tau - w_1^\alpha) + T_{33}^\alpha(w_3^\tau - w_3^\alpha) = (\partial_n w^\tau)|_3 - (\partial_n w^\alpha)|_3$$

Use that $w_1^\tau = w_1^\alpha = w_3^\alpha = 0$, and that $(\partial_n w^\alpha)|_3 = h_3^\alpha$ to get

$$(23) \quad T_{33}^\alpha w_3^\tau = (\partial_n w^\tau)|_3 - h_3^\alpha.$$

Analogously, we get

$$(24) \quad T_{33}^\beta w_3^\tau = (\partial_n w^\tau)|_3 - h_3^\beta.$$

Combine (23) and (24) to eliminate $(\partial_n w^\tau)|_3$ and obtain

$$(T_{33}^\alpha - T_{33}^\beta)w_3^\tau = -h_3^\alpha + h_3^\beta.$$

Observe that in effect, we can write the particular solution w^τ as

$$w^\tau(\mathbf{x}) = \begin{cases} w^\alpha(\mathbf{x}) + \hat{w}^\tau(\mathbf{x}) & \mathbf{x} \in \Omega^\alpha, \\ w^\beta(\mathbf{x}) + \hat{w}^\tau(\mathbf{x}) & \mathbf{x} \in \Omega^\beta, \end{cases}$$

The function w^τ must of course be smooth across Γ_3 , so the function \hat{w}^τ must have a jump that exactly offsets the discrepancy in the derivatives of w^α and w^β . This jump is precisely of size $h^\alpha - h^\beta$.

ALGORITHM 3 (Build stage for problems with body load)

This algorithms build all solution operators required to solve the non-homogeneous BVP (16). It is assumed that if node τ is a parent of node σ , then $\tau < \sigma$.

```

for  $\tau = N_{\text{boxes}}, N_{\text{boxes}} - 1, N_{\text{boxes}} - 2, \dots, 1$ 
  if ( $\tau$  is a leaf)
     $\mathbf{F}_{\text{c,ci}}^\tau = \begin{bmatrix} \mathbf{0} \\ \mathbf{A}_{\text{ci,ci}}^{-1} \end{bmatrix} \quad [pot.] \leftarrow [body \text{ load}]$ 
     $\mathbf{H}_{\text{ge,ci}}^\tau = \mathbf{D}_{\text{ge,c}} \mathbf{F}_{\text{c,ci}}^\tau \quad [deriv.] \leftarrow [body \text{ load}]$ 
     $\mathbf{S}_{\text{c,ge}}^\tau = \begin{bmatrix} \mathbf{I} \\ -\mathbf{A}_{\text{ci,ci}}^{-1} \mathbf{A}_{\text{ci,ce}} \end{bmatrix} \mathbf{I}_{\text{ce,ge}} \quad [pot.] \leftarrow [pot.]$ 
     $\mathbf{T}_{\text{ge,ge}}^\tau = \mathbf{D}_{\text{ge,c}} \mathbf{S}_{\text{c,ge}}^\tau \quad [deriv.] \leftarrow [pot.] \text{ (NfD operator)}$ 
  else
    Let  $\alpha$  and  $\beta$  be the children of  $\tau$ .
    Partition  $I_e^\alpha$  and  $I_e^\beta$  into vectors  $I_1, I_2$ , and  $I_3$  as shown in Figure 3.
     $\mathbf{X}_{\text{gi,gi}}^\tau = (\mathbf{T}_{3,3}^\alpha - \mathbf{T}_{3,3}^\beta)^{-1} \quad [pot.] \leftarrow [deriv.]$ 
     $\mathbf{S}_{\text{gi,ge}}^\tau = \mathbf{X}_{\text{gi,gi}}^\tau [-\mathbf{T}_{3,1}^\alpha \mid \mathbf{T}_{3,2}^\beta] \quad [pot.] \leftarrow [pot.]$ 
     $\mathbf{T}_{\text{ge,ge}}^\tau = \begin{bmatrix} \mathbf{T}_{1,1}^\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{2,2}^\beta \end{bmatrix} + \begin{bmatrix} \mathbf{T}_{1,3}^\alpha \\ \mathbf{T}_{2,3}^\beta \end{bmatrix} \mathbf{U}_{\text{gi,ge}}^\tau \quad [deriv.] \leftarrow [pot.] \text{ (NfD operator)}.$ 
  end if
end for

```

FIGURE 4. Build stage.

ALGORITHM 4 (Solver for problems with body load)

This program constructs an approximation \mathbf{u} to the solution u of (1). It assumes that all matrices required to represent the solution operator have already been constructed using Algorithm 3. It is assumed that if node τ is a parent of node σ , then $\tau < \sigma$.

Upwards pass — construct all particular solutions:

```

for  $\tau = N_{\text{boxes}}, N_{\text{boxes}} - 1, N_{\text{boxes}} - 2, \dots, 1$ 
  if ( $\tau$  is a leaf)
    # Compute the derivatives of the local particular solution.
     $\mathbf{h}_{\text{ge}}^\tau = \mathbf{H}_{\text{ge,ci}}^\tau \mathbf{g}_{\text{ci}}^\tau$ 
  else
    Let  $\alpha$  and  $\beta$  denote the children of  $\tau$ .
    # Compute the local particular solution.
     $\mathbf{w}_{\text{gi}}^\tau = \mathbf{X}_{\text{gi,gi}}^\tau (-\mathbf{h}_3^\alpha + \mathbf{h}_3^\beta)$ .
    # Compute the derivatives of the local particular solution.
     $\mathbf{h}_{\text{ge}}^\tau = \begin{bmatrix} \mathbf{h}_1^\alpha \\ \mathbf{h}_2^\beta \end{bmatrix} + \begin{bmatrix} \mathbf{T}_{1,3}^\alpha \\ \mathbf{T}_{2,3}^\beta \end{bmatrix} \mathbf{w}_{\text{gi}}^\tau$ .
  end if
end for

```

Downwards pass — construct all potentials:

```

# Use the provided Dirichlet data to set the solution on the exterior of the root.
 $\mathbf{u}(k) = f(\mathbf{x}_k)$  for all  $k \in I_e^1$ .
for  $\tau = 1, 2, 3, \dots, N_{\text{boxes}}$ 
  if ( $\tau$  is a parent)
    # Add the homogeneous term and the particular term.
     $\mathbf{u}_{\text{gi}}^\tau = \mathbf{U}_{\text{gi,ge}}^\tau \mathbf{u}_{\text{ge}}^\tau + \mathbf{w}_{\text{gi}}^\tau$ .
  else
    # Add the homogeneous term and the particular term.
     $\mathbf{u}_{\text{c}}^\tau = \mathbf{U}_{\text{c,ge}}^\tau \mathbf{u}_{\text{ge}}^\tau + \mathbf{F}_{\text{c,ci}}^\tau \mathbf{g}_{\text{ci}}^\tau$ .
  end
end for

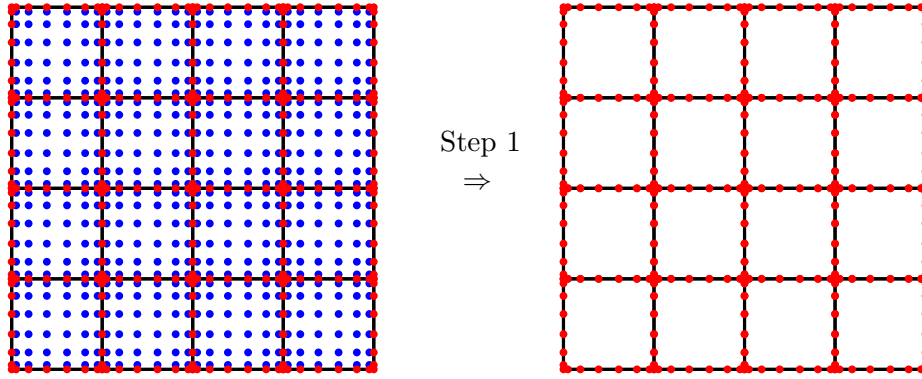
```

FIGURE 5. Solve stage.

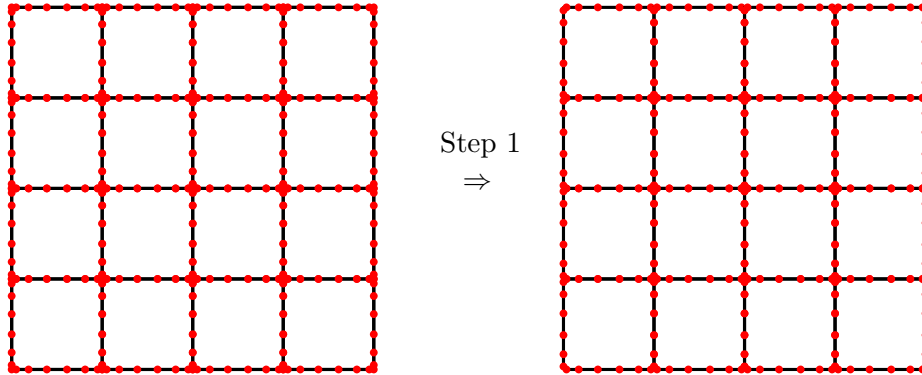
APPENDIX A. A GRAPHICAL ILLUSTRATION OF THE ALGORITHM

This section provides an illustrated overview of the hierarchical merge process described in detail in Section 5.1. The figures illustrate a situation in which a square domain $\Omega = [0, 1]^2$ is split into 4×4 leaf boxes on the finest level, and a 6×6 spectral grid is used in each leaf.

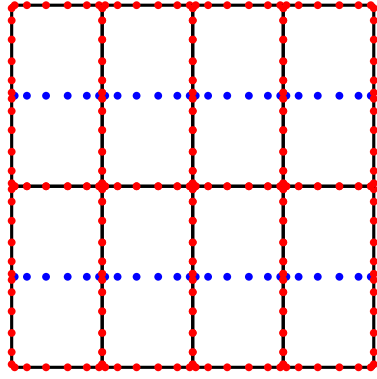
Step 1: Partition the box Ω into 16 small boxes that each holds an 6×6 Cartesian mesh of Chebyshev nodes. For each box, identify the internal nodes (marked in blue) and eliminate them as described in Section 3. Construct the solution operator \mathbf{U} , and the DtN operators encoded in the matrices \mathbf{V} and \mathbf{W} .



Step 2: Switch tabulation points on the boundary from Chebyshev nodes to Legendre nodes. The main purpose is to remove the corner nodes.

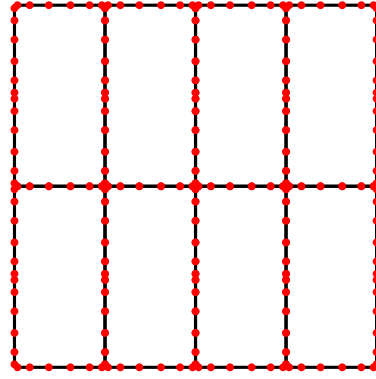


Step 3: Merge the small boxes by pairs as described in Section 4. The equilibrium equation for each rectangle is formed using the DtN operators of the two small squares it is made up of. The result is to eliminate the interior nodes (marked in blue) of the newly formed larger boxes. Construct the solution operator \mathbf{U} and the DtN matrices \mathbf{V} and \mathbf{W} for the new boxes.

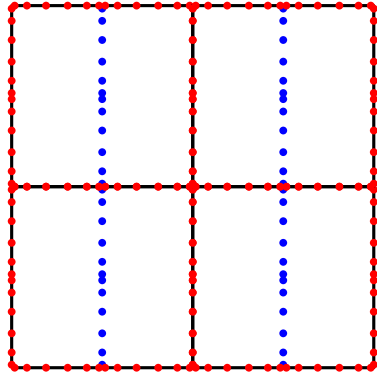


Step 2

\Rightarrow

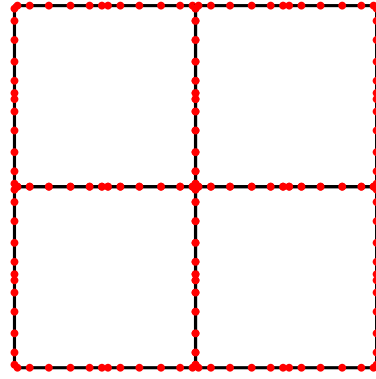


Step 4: Merge the boxes created in Step 3 in pairs, again via the process described in Section 4.

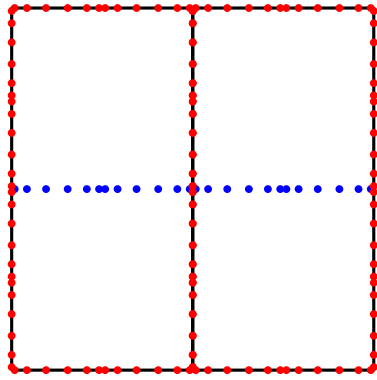


Step 3

\Rightarrow

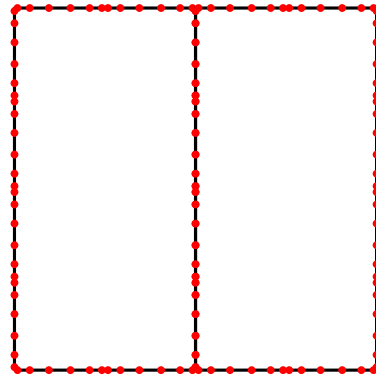


Step 5: Repeat the merge process once more.

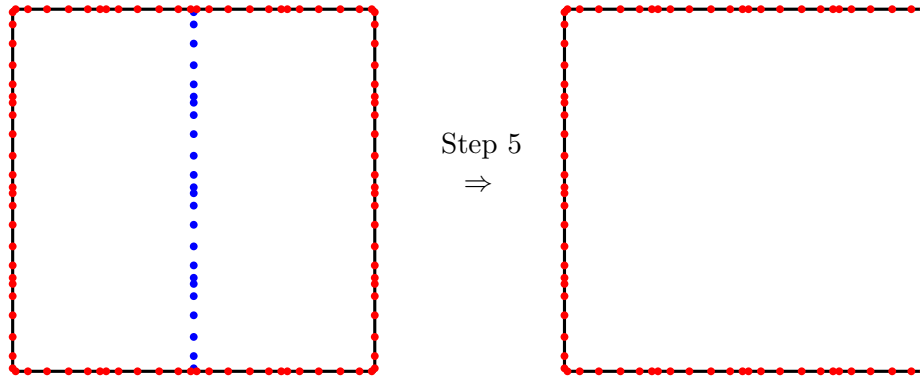


Step 4

\Rightarrow



Step 6: Repeat the merge process one final time to obtain the DtN operator for the boundary of the whole domain.



REFERENCES

- [1] Mario Bebendorf, *Hierarchical matrices*, Lecture Notes in Computational Science and Engineering, vol. 63, Springer-Verlag, Berlin, 2008, A means to efficiently solve elliptic boundary value problems. MR 2451321 (2009k:15001)
- [2] Steffen Börm, *Efficient numerical methods for non-local operators*, EMS Tracts in Mathematics, vol. 14, European Mathematical Society (EMS), Zürich, 2010, \mathcal{H}^2 -matrix compression, algorithms and analysis. MR 2767920
- [3] Timothy A Davis, *Direct methods for sparse linear systems*, vol. 2, Siam, 2006.
- [4] I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct methods for sparse matrices*, Oxford, 1989.
- [5] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. on Numerical Analysis **10** (1973), 345–363.
- [6] A. Gillman and P. Martinsson, *A direct solver with $o(n)$ complexity for variable coefficient elliptic pdes discretized via a high-order composite spectral collocation method*, SIAM Journal on Scientific Computing **36** (2014), no. 4, A2023–A2046, arXiv.org report #1307.2665.
- [7] Adrianna Gillman, AlexH. Barnett, and Per-Gunnar Martinsson, *A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media*, BIT Numerical Mathematics **55** (2015), no. 1, 141–170 (English).
- [8] W. Hackbusch, B. Khoromskij, and S. Sauter, *On \mathcal{H}^2 -matrices*, Lectures on Applied Mathematics, Springer Berlin, 2002, pp. 9–29.
- [9] Wolfgang Hackbusch, *A sparse matrix arithmetic based on H -matrices; Part I: Introduction to H -matrices*, Computing **62** (1999), 89–108.
- [10] T.S. Haut, T. Babb, P.G. Martinsson, and B.A. Wingate, *A high-order scheme for solving wave propagation problems via the direct construction of an approximate time-evolution operator*, arXiv preprint arXiv:1402.5168 (2014).
- [11] J.S. Hesthaven, P.G. Dinesen, and J.P. Lynov, *Spectral collocation time-domain modeling of diffractive optical elements*, Journal of Computational Physics **155** (1999), no. 2, 287 – 306.
- [12] P.G. Martinsson, *A composite spectral scheme for variable coefficient helmholtz problems*, arXiv preprint arXiv:1206.4136 (2012).
- [13] P.G. Martinsson, *A direct solver for variable coefficient elliptic pdes discretized via a composite spectral collocation method*, Journal of Computational Physics **242** (2013), no. 0, 460 – 479.
- [14] H.P. Pfeiffer, L.E. Kidder, M.A. Scheel, and S.A. Teukolsky, *A multidomain spectral method for solving elliptic equations*, Computer physics communications **152** (2003), no. 3, 253–273.
- [15] L.N. Trefethen, *Spectral methods in matlab*, SIAM, Philadelphia, 2000.